GALAHAD GLS

USER DOCUMENTATION GALAHAD Optimization Library version 5.1

1 SUMMARY

`GALAHAD_GLS` is a suite of Fortran 95 procedures for solving sparse unsymmetric system of linear equations. Given a sparse matrix $\mathbf{A} = \{a_{ij}\}_{m \times n}$, this subroutine solves the system $\mathbf{Ax} = \mathbf{b}$ (or optionally $\mathbf{A}^T\mathbf{x} = \mathbf{b}$). The matrix $\mathbf{A}$ can be rectangular.

This Fortran 95 code offers additional features to the Fortran 77 HSL code `MA33` which it calls. The storage required for the factorization is chosen automatically and, if there is insufficient space for the factorization, more space is allocated and the factorization is repeated. The package also returns the number of entries in the factors and has facilities for identifying the rows and columns that are treated specially when the matrix is singular or rectangular.

ATTRIBUTES — Versions: `GALAHAD_GLS_single`, `GALAHAD_GLS_double`. **Remark:** `GALAHAD_GLS` is a Fortran 95 encapsulation of the core subroutines from the HSL Fortran 77 package `MA33` and offers some additional facilities. The user interface is designed to be equivalent to a subset of the more recent HSL package `HSL_MA48`, so many features of the two packages may be used interchangeably. **Uses:** `GALAHAD_SMT`, `MA33` from HSL. **Date:** March 2006. **Origin:** Interface by N. I. M. Gould, Rutherford Appleton Laboratory, documentation follows that of I.S. Duff and J.K. Reid, ibid. **Language:** Fortran 95.

2 HOW TO USE THE PACKAGE

The package is available with single, double and (if available) quadruple precision reals, and either 32-bit or 64-bit integers. Access to the 32-bit integer, single precision version requires the `USE` statement

```
        USE GALAHAD_GLS_single
```

with the obvious substitution `GALAHAD_GLS_double`, `GALAHAD_GLS_quadruple`, `GALAHAD_GLS_single_64`, `GALAHAD_GLS_double_64` and `GALAHAD_GLS_quadruple_64` for the other variants.

If it is required to use more than one of the modules at the same time, the derived types (Section 2.2) and the subroutines (Section 2.3) must be renamed on one of the `USE` statements.

There are four principal subroutines for user calls:

1. The subroutine `GLS_INITIALIZE` must be called to initialize the structure for the factors. It may also be called to set default values for the components of the control structure. If non-default values are wanted for any of the control components, the corresponding components should be altered after the call to `GLS_INITIALIZE`.

2. `GLS_ANALYSE` accepts the pattern of A and chooses pivots for Gaussian elimination using a selection criterion to preserve sparsity. It will optionally find an ordering to block triangular form and exploit that structure. An option exists to restrict pivoting to the diagonal, which might reduce fill-in and operations if the matrix has a symmetric structure. It is possible to perform an analysis without generating the factors, in which case data on the costs of a subsequent factorization are returned to the user. It is also possible to request that a set of columns are pivoted on last in which case a subsequent factorization can avoid factorization operations on the earlier columns.

3. `GLS_SOLVE` uses the factors generated by `GLS_ANALYSE` to solve a system of equations $\mathbf{Ax} = \mathbf{b}$ or $\mathbf{A}^T\mathbf{x} = \mathbf{b}$.

4. `GLS_FINALIZE` reallocates the arrays held inside the structure for the factors to have size zero. It should be called when all the systems involving its matrix have been solved unless the structure is about to be used for the factors of another matrix.

There is an auxiliary subroutine for user calls after a successful factorization:

5. GLS_SPECIAL_ROWS_AND_COLS identifies the rows and columns that are treated specially when the matrix is singular or rectangular. It is for use following a call of GLS_ANALYSE.

2.1 Real and integer kinds

We use the terms integer and real to refer to the fortran keywords REAL(rp_) and INTEGER(ip_), where rp_ and ip_ are the relevant kind values for the real and integer types employed by the particular module in use. The former are equivalent to default REAL for the single precision versions, DOUBLE PRECISION for the double precision cases and quadruple-precision if 128-bit reals are available, and correspond to rp_ = real32, rp_ = real64 and rp_ = real128 respectively as defined by the fortran iso_fortran_env module. The latter are default (32-bit) and long (64-bit) integers, and correspond to ip_ = int32 and ip_ = int64, respectively, again from the iso_fortran_env module.

2.2 The derived data types

Six derived data types are accessible from the package.

2.2.1 The derived data type for holding the matrix

The derived data type SMT_TYPE is used to hold the matrix. The components of SMT_TYPE used are:

M is an INTEGER(ip_) scalar which holds the number of rows m of the matrix A. **Restriction:** M $\geq$ 1.

N is an INTEGER(ip_) scalar which holds the number of columns n of the matrix A. **Restriction:** N $\geq$ 1.

NE is an INTEGER(ip_) scalar which holds the number of matrix entries. **Restriction:** NE $\geq$ 0.

VAL is a REAL(rp_) pointer array of length at least NE, the leading part of which holds the values of the entries. Duplicate entries are summed.

ROW is an INTEGER(ip_) pointer array of length at least NE, the leading part of which holds the row indices of the entries.

COL is an INTEGER(ip_) pointer array of length at least NE, the leading part of which holds the column indices of the entries.

2.2.2 The derived data type for holding control parameters

The derived data type GLS_CONTROL is used to hold controlling data. Default values may be obtained by calling GLS_initialize (see Section 2.3.1). The components of GLS_CONTROL are:

LP is an INTEGER(ip_) scalar used by the subroutines as the output unit for error messages. If it is negative, these messages will be suppressed. The default value is 6.

WP is an INTEGER(ip_) scalar used by the subroutines as the output unit for warning messages. If it is negative, these messages will be suppressed. The default value is 6.

MP is an INTEGER(ip_) scalar used by the subroutines as the output unit for diagnostic printing. If it is negative, these messages will be suppressed. The default value is 6.

LDIAG is an INTEGER(ip_) scalar used by the subroutines to control diagnostic printing. If LDIAG is less than 1, no messages will be output. If the value is 1, only error messages will be printed. If the value is 2, then error and warning messages will be printed. If the value is 3, scalar data and a few entries of array data on entry and exit from each subroutine will be printed. If the value is greater than 3, all data will be printed on entry and exit. This output comes from the Fortran 77 MA33 routines called by GALAHAD_GLS. The default value is 2.

LA is an INTEGER(ip_) scalar used by GLS_ANALYSE. LA is set to FILL_IN $*$ NE by GLS_ANALYSE. The default for FILL_IN is 3 but, if the user knows that there may be significant fill-in during factorization, it may be efficient to increase this value.

MAXLA is an INTEGER(ip_) scalar used by GLS_ANALYSE. An error return occurs if the real array that holds data for the factors is too small and reallocating it to have size changed by the factor MULTIPLIER would make its size greater than MAXLA. The default value is HUGE(0).

MULTIPLIER is a REAL(rp_) scalar used by GLS_ANALYSE when a real or integer array that holds data for the factors is too small. The array is reallocated with its size changed by the factor MULTIPLIER. The default value is 2.0.

REDUCE is a REAL(rp_) scalar that reduces the size of previously allocated internal workspace arrays if they are larger than currently required by a factor of REDUCE or more. The default value is 2.0.

SWITCH is an REAL(rp_) scalar used by GLS_ANALYSE to control the switch from sparse to full matrix processing when factorizing the diagonal blocks. The switch is made when the ratio of number of entries in the reduced matrix to the number that it would have as a full matrix is greater than SWITCH. A value greater than 1.0 is treated as 1.0. The default value is 0.5.

U is a REAL(rp_) scalar that is used by GLS_ANALYSE. It holds the threshold parameter for the pivot control. The default value is 0.01. For problems requiring greater than average numerical care a higher value than the default would be advisable. Values greater than 1.0 are treated as 1.0 and less than 0.0 as 0.0.

DROP is a REAL(rp_) scalar that is used by GLS_ANALYSE. Any entry whose modulus is less than DROP will be dropped from the factorization. The factorization will then require less storage but will be inaccurate. The default value is 0.0.

TOLERANCE is a REAL(rp_) scalar that is used by GLS_ANALYSE. If it is set to a positive value, any pivot whose modulus is less than TOLERANCE will be treated as zero. the factorization. The default value is 0.0.

CGCE is a REAL(rp_) scalar that is used by GLS_SOLVE. It is used to monitor the convergence of the iterative refinement. If successive corrections do not decrease by a factor of at least CGCE, convergence is deemed to be too slow and GLS_SOLVE terminates with SINFO%FLAG set to -8. The default value is 0.5.

PIVOTING is a INTEGER(ip_) scalar that is used to control numerical pivoting by GLS_ANALYSE. If PIVOTING has a positive value, each pivot search is limited to a maximum of PIVOTING columns. If PIVOTING is set to the value 0, a full Markowitz search technique is used to find the best pivot. This is usually only a little slower, but can occasionally be very slow. It may result in reduced fill-in. The default value is 3.

DIAGONAL_PIVOTING is a LOGICAL scalar used by GLS_ANALYSE to limit pivoting to the diagonal. It will do so if DIAGONAL_PIVOTING is set to .TRUE.. Its default value is .FALSE..

FILL_IN is an INTEGER(ip_) scalar used by GLS_ANALYSE to determine the initial storage allocation for the matrix factors. It will be set to FILL_IN times the value of MATRIX%NE. The default value is 3.

BTF is an INTEGER(ip_) scalar used by GLS_ANALYSE to define the minimum size of a block of the block triangular form other than the final block. If block triangularization is not wanted, BTF should be set to a value greater than or equal to MAX(M,N). Block triangulation will only be attempted for square (M $=$ N) matrices. A non-positive value is regarded as the value 1. For further discussion of this variable, see Section 2.6. The default value is 1.

STRUCT is a LOGICAL scalar used by GLS_ANALYSE. If STRUCT is set to .TRUE., the subroutine will exit immediately structural singularity is detected. The default value is .FALSE..

FACTOR_BLOCKING is an INTEGER(ip_) scalar used by GLS_ANALYSE to determine the block size used for the Level 3 BLAS within the full factorization. If it is set to 1, Level 1 BLAS is used, if to 2, Level 2 BLAS is used. The default value is 32.

SOLVE_BLAS is an INTEGER(ip_) scalar used by GLS_SOLVE to determine whether Level 2 BLAS is used (SOLVE_BLAS > 1) or not (SOLVE_BLAS ≤ 1). The default value is 2.

MAXIT is an INTEGER(ip_) scalar used by GLS_SOLVE to limit the number of refinement iterations. If MAXIT is set to zero then GLS_SOLVE will not perform any error analysis or iterative refinement. The default value is 10.

2.2.3 The derived data type for holding informational parameters from the analysis phase

The derived data type GLS_AINFO is used to hold parameters that give information about the progress and needs of the analysis phase of the algorithm. The components of GLS_AINFO are:

FLAG is an INTEGER(ip_) scalar. The value zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.1.

MORE is an INTEGER(ip_) scalar that provides further information in the case of an error, see Section 2.4.1.

OOR is an INTEGER(ip_) scalar which is set to the number of entries with one or both indices out of range.

DUP is an INTEGER(ip_) scalar which is set to the number of duplicate entries.

DROP is an INTEGER(ip_) scalar which is set to the number of entries dropped from the data structure.

STAT is an INTEGER(ip_) scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

OPS is a REAL(rp_) scalar which is set to the number of floating-point operations required by the factorization.

RANK is an INTEGER(ip_) scalar that gives an estimate of the rank of the matrix.

STRUC_RANK is an INTEGER(ip_) scalar that, if BTF is less than or equal to N, holds the structural rank of the matrix. If BTF $>$ N, STRUC_RANK is set to min(M, N).

LEN_ANALYSE is an INTEGER(ip_) scalar that gives the number of REAL(rp_) and INTEGER(ip_) words required for the analysis.

LEN_FACTORIZE is an INTEGER(ip_) scalar that gives the number of REAL(rp_) and INTEGER(ip_) words required for successful subsequent factorization assuming the same pivot sequence and set of dropped entries can be used.

NCMPA is an INTEGER(ip_) scalar that holds the number of compresses of the internal data structure performed by GLS_ANALYSE. If NCMPA is fairly large (say greater than 10), performance may be very poor.

LBLOCK is an INTEGER(ip_) scalar that holds the order of the largest non-triangular block on the diagonal of the block triangular form. If the matrix is rectangular, LBLOCK will hold the number of rows.

SBLOCK is an INTEGER(ip_) scalar that holds the sum of the orders of all the non-triangular blocks on the diagonal of the block triangular form. If the matrix is rectangular, SBLOCK will hold the number of columns.

TBLOCK is an INTEGER(ip_) scalar that holds the total number of entries in all the non-triangular blocks on the diagonal of the block triangular form.

2.2.4 The derived data type for holding informational parameters from the factorization phase

The derived data type GLS_FINFO is used to hold parameters that give information about the progress and needs of the factorization phase of the algorithm. The components of GLS_FINFO are:

FLAG is an INTEGER(ip_) scalar. The value zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.2.

MORE is an INTEGER(ip_) scalar that provides further information in the case of an error, see Section 2.4.2.

STAT is an INTEGER(ip_) scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

OPS is a REAL(rp_) scalar which is set to the number of floating-point operations required by the factorization.

DROP is an INTEGER(ip_) scalar which is set to the number of entries dropped from the data structure.

LEN_FACTORIZE is an INTEGER(ip_) scalar that gives the number of REAL(rp_) and INTEGER(ip_) words required for successful subsequent factorization assuming the same pivot sequence and set of dropped entries can be used.

SIZE_FACTOR is an INTEGER(ip_) scalar that gives the number of entries in the matrix factors.

RANK is an INTEGER(ip_) scalar that gives an estimate of the rank of the matrix.

2.2.5 The derived data type for holding informational parameters from the solution phase

The derived data type GLS_SINFO is used to hold parameters that give information about the progress and needs of the solution phase of the algorithm. The components of GLS_SINFO are:

FLAG is an INTEGER(ip_) scalar. The value zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.3.

MORE is an INTEGER(ip_) scalar that provides further information in the case of an error, see Section 2.4.3.

STAT is an INTEGER(ip_) scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

2.2.6 The derived data type for the factors of a matrix

The derived data type GLS_FACTORS is used to hold the factors of a matrix, and has private components.

2.3 Argument lists and calling sequences

We use square brackets [ ] to indicate OPTIONAL arguments.

2.3.1 The initialization subroutine

The initialization subroutine must be called for each structure used to hold the factors. It may also be called for a structure used to control the subroutines. Each argument is optional. A call with no arguments has no effect.

```
CALL GLS_initialize( ([FACTORS] [,CONTROL])
```

FACTORS is optional, scalar, of INTENT(OUT) and of type GLS_FACTORS. On exit, its pointer array components will be null. Without such initialization, these components are undefined and other calls are likely to fail.

CONTROL is optional, scalar, of INTENT(OUT) and of type GLS_CONTROL. On exit, its components will have been given the default values specified in Section 2.2.2.

2.3.2 To analyse the sparsity pattern and factorize the matrix

```
CALL GLS_ANALYSE( MATRIX, FACTORS, CONTROL, AINFO, FINFO )
```

MATRIX is scalar, of INTENT(IN) and of type SMT_TYPE. The user must set the components M, N, NE, ROW, COL, and VAL, and they are not altered by the subroutine. **Restrictions:** MATRIX%M ≥ 1, MATRIX%N ≥ 1, and MATRIX%NE ≥ 0.

FACTORS is scalar, of INTENT(INOUT) and of type GLS_FACTORS. It must have been initialized by a call to GLS_INITIALIZE or have been used for a previous calculation. In the latter case, the previous data will be lost but the pointer arrays will not be reallocated unless they are found to be too small.

CONTROL is scalar, of INTENT(IN) and of type GLS_CONTROL. Its components control the action, as explained in Section 2.2.2.

AINFO is scalar, of INTENT(OUT) and of type GLS_AINFO. Its components provide information about the execution, as explained in Section 2.2.3.

FINFO is scalar, of INTENT(OUT) and of type GLS_FINFO. If present, the call to GLS_ANALYSE will compute and store the factorization of the matrix. Its components provide information about the execution of the factorization, as explained in Section 2.2.4.

2.3.3 To solve a set of equations

```
CALL GLS_SOLVE( MATRIX, FACTORS, RHS, X, CONTROL, SINFO[, TRANS] )
```

MATRIX is scalar, of INTENT(IN) and of type SMT_TYPE. It must be unaltered since the call to GLS_ANALYSE and is not altered by the subroutine.

FACTORS is scalar, of INTENT(IN) and of type GLS_FACTORS. It must be unaltered since the call to GLS_ANALYSE and is not altered by the subroutine.

RHS is an array of shape (n) of INTENT(IN) and of type REAL(rp_). It must be set by the user to the vector **b**.

X is an array of shape (n) of INTENT(OUT) and of type REAL(rp_). On return it holds the solution **x**.

CONTROL is scalar, of INTENT(IN) and of type GLS_CONTROL. Its components control the action, as explained in Section 2.2.2.

SINFO is scalar, of INTENT(OUT) and of type GLS_SINFO. Its components provide information about the execution, as explained in Section 2.2.5.

TRANS is scalar, optional, of INTENT(IN) and of type INTEGER(ip_). If present $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ is solved, otherwise the solution is obtained for $\mathbf{A}\mathbf{x} = \mathbf{b}$.

2.3.4 The termination subroutine

All previously allocated arrays are deallocated as follows:

```
CALL GLS_FINALIZE( FACTORS, CONTROL, INFO )
```

FACTORS is scalar, of INTENT(INOUT) and of type GLS_FACTORS. On exit, its pointer array components will have been deallocated. Without such finalization, the storage occupied is unavailable for other purposes. In particular, this is very wasteful if the structure goes out of scope on return from a procedure.

CONTROL is scalar, of INTENT(IN) and of type GLS_CONTROL. Its components control the action, as explained in Section 2.2.2.

INFO is scalar, of INTENT(OUT) and of type INTEGER(ip_). On return, the value 0 indicates success. Any other value is the STAT value of an ALLOCATE or DEALLOCATE statement that has failed.

2.3.5 To identify the rows and columns that are treated specially following a successful factorization

```
CALL GLS_SPECIAL_ROWS_AND_COLS( FACTORS, RANK, ROWS, COLS, INFO )
```

FACTORS is scalar, of INTENT(IN) and of type GLS_FACTORS. It must be unaltered since the call to GLS_ANALYSE and is not altered by the subroutine.

RANK is an INTEGER(ip_) variable that need not be set by the user. On return, it holds the calculated rank of the matrix (it is the rank of the matrix actually factorized).

ROWS is an INTEGER(ip_) array of length M that need not be set by the user. On return, it holds a permutation. The indices of the rows that are taken into account when solving $\mathbf{Ax} = \mathbf{b}$ are ROWS(i), $i \leq$ RANK.

COLS is an INTEGER(ip_) array of length N that need not be set by the user. On return, it holds a permutation. The indices of the columns that are taken into account when solving $\mathbf{Ax} = \mathbf{b}$ are COLS(j), $j \leq$ RANK.

INFO is an INTEGER(ip_) variable that need not be set by the user. On return, its value is 0 if the call was successful, -1 if the allocation of a temporary array failed, or -2 if the subsequent deallocation failed.

2.4 Warning and error messages

2.4.1 When performing the analysis

A successful return from the analysis phase within GLS_ANALYSE is indicated by AINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. Possible negative values are:

-1 Value of MATRIX%M out of range. MATRIX%M < 1. AINFO%MORE is set to value of MATRIX%M.

-2 Value of MATRIX%N out of range. MATRIX%N < 1. AINFO%MORE is set to value of MATRIX%N.

-3 Value of MATRIX%NE out of range. MATRIX%NE < 0. AINFO%MORE is set to value of MATRIX%NE.

-4 Failure of an allocate or deallocate statement. AINFO%STAT is set to the STAT value.

-5 On a call with STRUCT having the value .TRUE., the matrix is structurally rank deficient. The structural rank is given by STRUC_RANK.

A positive flag value is associated with a warning message which will be output on unit AINFO%WP. Possible positive values are:

1 Index (in MATRIX%ROW or MATRIX%COL) out of range. Action taken by subroutine is to ignore any such entries and continue. AINFO%OOR is set to the number of such entries. Details of the first ten are optionally printed on unit CONTROL%MP.

2 Duplicate indices. Action taken by subroutine is to sum corresponding reals. AINFO%DUP is set to the number of duplicate entries. Details of the first ten are optionally printed on unit CONTROL%MP.

3 Combination of a 1 and a 2 warning.

4 The matrix is rank deficient with estimated rank AINFO%RANK.

5 Combination of a 1 and a 4 warning.

6 Combination of a 2 and a 4 warning.

7 Combination of a 1, a 2, and a 4 warning.

16 More space required than initial allocation. Size of LA used is given in AINFO%MORE.

17 to 23 Combination of warnings that sum to this total.

2.4.2 When factorizing the matrix

A successful return from the factorization phase within GLS_ANALYSE is indicated by FINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. In this case, no solution will have been calculated. Possible negative values are:

-1 Value of MATRIX%M differs from the GLS_ANALYSE value. FINFO%MORE holds value of MATRIX%M.

-2 Value of MATRIX%N differs from the GLS_ANALYSE value. FINFO%MORE holds value of MATRIX%N.

-3 Value of MATRIX%NE out of range. MATRIX%NE < 0. FINFO%MORE holds value of MATRIX%NE.

-4 Failure of an allocate or deallocate statement. FINFO%STAT is set to the STAT value.

-7 The real array that holds data for the factors needs to be bigger than CONTROL%MAXLA.

-10 GLS_FACTORIZE has been called without a prior call to GLS_ANALYSE.

A positive flag value is associated with a warning message which will be output on unit CONTROL%MP. In this case, a factorization will have been calculated.

4 Matrix is rank deficient. In this case, FINFO%RANK will be set to the rank of the factorization. In the subsequent solution, all columns in the singular block will have the corresponding component in the solution vector set to zero.

16 More space required than initial allocation. Size of LA used is given in FINFO%MORE.

20 Combination of a 4 and a 16 warning.

2.4.3 When using factors to solve equations

A successful return from GLS_SOLVE is indicated by SINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. In this case, the solution will not have been completed. Possible negative values are:

-1 Value of MATRIX%M differs from the GLS_ANALYSE value. SINFO%MORE holds value of MATRIX%M.

-2 Value of MATRIX%N differs from the GLS_ANALYSE value. SINFO%MORE holds value of MATRIX%N.

-3 Value of MATRIX%NE out of range. MATRIX%NE < 0. SINFO%MORE holds value of MATRIX%NE.

-10 GLS_SOLVE has been called without a prior call to GLS_ANALYSE.

2.5 Rectangular and rank deficient matrices

Rectangular matrices are handled by the code although no attempt is made at prior block triangularization. Rank deficient matrices are also factorized and a warning flag is set (`AINFO%FLAG` or `FINFO%FLAG` set to `4`). If `CONTROL%STRUCT` is set to `.TRUE.`, then an error return occurs (`AINFO%FLAG` = `-5`) if block triangularization is attempted and the matrix is structurally singular.

The package identifies a square submatrix of $\mathbf{A}$ that it considers to be nonsingular. When solving $\mathbf{Ax} = \mathbf{b}$, equations outside this submatrix are ignored and solution components that correspond to columns outside the submatrix are set to zero. `GLS_SPECIAL_ROWS_AND_COLS` identifies the rows and columns of this submatrix from stored integer data.

It should be emphasized that the primary purpose of the package is to solve square nonsingular sets of equations. The rank is determined from the number of pivots that are not small or zero. There are more reliable (but much more expensive) ways of determining numerical rank.

2.6 Block upper triangular form

Many large unsymmetric matrices can be permuted to the form

$$\mathbf{PAQ} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdot & \cdot & \cdot & \cdot \\ & \mathbf{A}_{22} & \cdot & \cdot & \cdot & \cdot \\ & & \mathbf{A}_{33} & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot \\ & & & & & \mathbf{A}_{\ell\ell} \end{pmatrix}$$

whereupon the system

$$\mathbf{Ax} = \mathbf{b} \;\; (\text{or } \; \mathbf{A}^T\mathbf{x} = \mathbf{b})$$

can be solved by block back-substitution giving a saving in storage and execution time if the matrices $\mathbf{A}_{ii}$ are much smaller than $\mathbf{A}$.

Since it is not very efficient to process a small block (for example a 1 by 1 block), any block of size less than `CONTROL%BTF` other than the final block is merged with its successor.

2.7 Badly-scaled systems

If the user's input matrix has entries differing widely in magnitude, then an inaccurate solution may be obtained. In such cases, the user is advised to first use the HSL package `MC29A/AD` to obtain scaling factors for the matrix and then explicitly scale it prior to calling this package.

3 GENERAL INFORMATION

Use of common: None.

Workspace: Provided automatically by the module.

Other routines called directly: `MC13E/ED`, `MC20A/AD`, `MC21B/BD`, `MA33A/AD`, `MA33C/CD`.

Other modules used directly: `GALAHAD_SMT_single/double`.

Input/output: Error, warning and diagnostic messages only. Error messages on unit `CONTROL%LP` and warning and diagnostic messages on unit `CONTROL%WP` and `CONTROL%MP`, respectively. These have default value `6`, and printing of these messages is suppressed if the relevant unit number is set negative. These messages are also suppressed if `GLS_CONTROL%LDIAG` is less than `1`.

Restrictions: `MATRIX%M` ≥ 1, `MATRIX%N` ≥ 1, `MATRIX%NE` ≥ 0.

Portability: ISO Fortran 95.

4 METHOD

A version of sparse Gaussian elimination is used. Subroutine `GLS_ANALYSE` calls `MA33A/AD` to compute a pivot ordering for the decomposition of A into sparse LU factors. Pivoting is used to preserve sparsity in the factors. Each pivot a_{pj} is required to satisfy a stability test

$$|a_{pj}| \geq u \max_i |a_{ij}|$$

within the reduced matrix, where u is the threshold held in `CONTROL%U`, with default value `0.01`. The subroutine then computes the numerical factors based on the chosen pivot order. Subroutine `GLS_SOLVE` uses the factors found by `GLS_ANALYSE` to solve systems of equations by calling `MA33C/CD`.

A discussion of the design of the predecessor to the `MA33` routines called by this package is given by Duff and Reid, *ACM Trans Math Software* **5**, 1979, pp 18-35.

5 EXAMPLE OF USE

We illustrate the use of the package on the solution of the set of equations

$$\begin{pmatrix} 11 & 12 & \\ 21 & 22 & 23 \\ & 32 & 33 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 23 \\ 66 \\ 65 \end{pmatrix}$$

and a second set

$$\begin{pmatrix} 11 & 21 & \\ 12 & 22 & 32 \\ & 23 & 33 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 32 \\ 66 \\ 56 \end{pmatrix}$$

involving the transpose (note that this example does not illustrate all the facilities). Then we may use the following code:

```
   PROGRAM GALAHAD_GLS_example  ! GALAHAD 4.1 - 2022-11-25 AT 09:20 GMT.
   USE GALAHAD_GLS_DOUBLE
   IMPLICIT NONE
   INTEGER, PARAMETER :: wp = KIND( 1.0D+0 )
   INTEGER :: info, rank
   INTEGER, PARAMETER :: m = 3
   INTEGER, PARAMETER :: n = 3
   INTEGER, PARAMETER :: ne = 7
   TYPE ( SMT_TYPE ) :: matrix
   TYPE ( GLS_CONTROL ) :: control
   TYPE ( GLS_AINFO ) :: ainfo
   TYPE ( GLS_FINFO ) :: finfo
   TYPE ( GLS_SINFO ) :: sinfo
   TYPE ( GLS_FACTORS ) :: factors
   INTEGER :: ROWS( m ), COLS( n )
   REAL ( KIND = wp ) :: X( n ), B( m )
! record matrix order and number of entries
   matrix%m = m ; matrix%n = n ; matrix%ne = ne
! allocate and set matrix
!  CALL SMT_put( matrix%type, 'COORDINATE', info )   ! Specify co-ordinate
   ALLOCATE( matrix%val( ne ),  matrix%row( ne ),  matrix%col( ne ) )
   matrix%row( : ne ) = (/ 1, 2, 3, 2, 1, 3, 2 /)
   matrix%col( : ne ) = (/ 1, 3, 3, 1, 2, 2, 2 /)
   matrix%val( : ne ) = (/ 11.0_wp, 23.0_wp, 33.0_wp, 21.0_wp, 12.0_wp,        &
                           32.0_wp, 22.0_wp /)
! initialize structures
   CALL GLS_initialize( factors, control )
! analyse and factorize
   CALL GLS_analyse( matrix, factors, control, ainfo, finfo )
   IF ( AINFO%flag < 0 ) THEN
     WRITE( 6, "( ' Failure of GLS_ANALYSE with AINFO%flag = ',  I0 )" )       &
       AINFO%flag
     STOP
   END IF
! write row and column reorderings
   CALL GLS_special_rows_and_cols( factors, rank, ROWS, COLS, info )
   WRITE( 6, "( ' row orderings:', /, ( 10I5 ) )" ) ROWS( : rank )
   WRITE( 6, "( ' column orderings:', /, ( 10I5 ) )" ) COLS( : rank )
! set right-hand side and solve system
   B = (/ 23.0_wp, 66.0_wp, 65.0_wp /)
   CALL GLS_solve( matrix, factors, B, X, control, sinfo )
   IF ( SINFO%flag == 0 ) WRITE( 6,                                            &
     "( ' Solution of set of equations without refinement is', /,              &
        & ( 6ES11.3 ) )" ) X
! now solve the transposed system
   B = (/ 32.0_wp, 66.0_wp, 56.0_wp /)
   CALL GLS_solve( matrix, factors, B, X, control, sinfo, trans = 1 )
   IF ( SINFO%flag == 0 ) WRITE( 6,                                            &
     "( ' Solution of set of transposed equations without refinement is', /,   &
        & ( 6ES11.3 ) )" ) X
! clean up
   CALL GLS_FINALIZE( factors, control, info )
   DEALLOCATE( matrix%val, matrix%row, matrix%col )
   STOP
   END PROGRAM GALAHAD_GLS_example
```

This produces the following output:

```
 row orderings:
    1    2    3
 column orderings:
    1    2    3
 Solution of set of equations without refinement is
  1.000E+00  1.000E+00  1.000E+00
 Solution of set of transposed equations without refinement is
  1.000E+00  1.000E+00  1.000E+00
```